Rapport sur le développement d'un programme d'alarme en Python

✅ Introduction

La programmation d'un système d'alarme constitue un excellent exercice pour appliquer des notions fondamentales en Python telles que la gestion du temps, les conditions logiques, les boucles et l'interaction avec l'utilisateur. Ce type de programme peut être utilisé pour des rappels, des notifications ou même comme base pour des systèmes plus avancés (domotique, sécurité, etc.). L'objectif de ce rapport est de présenter la conception, la logique interne, les difficultés rencontrées et les solutions adoptées lors du développement d'un programme d'alarme simple mais fonctionnel.

✅ Objectif du programme

Le programme d'alarme vise à :

- Permettre à l'utilisateur de définir une heure précise pour déclencher une alarme.
- Surveiller en continu l'heure actuelle du système.
- Déclencher une action (son, message, ou autre) lorsque l'heure programmée est atteinte.
- Assurer une utilisation simple, intuitive et fiable.

En résumé, il s'agit de créer un outil automatisé capable de notifier l'utilisateur au moment souhaité.

✅ Algorithme logique utilisé

Voici la logique générale du programme, présentée sous forme d'algorithme :

1. **Demander à l'utilisateur** de saisir l'heure de l'alarme (format HH:MM:SS).
2. **Valider** que le format est correct.
3. **Entrer dans une boucle infinie** qui vérifie l'heure actuelle.
4. **Comparer** l'heure du système avec l'heure programmée.
5. Si les deux correspondent :
    - Déclencher l'alarme (son, message, etc.).
    - Sortir de la boucle ou proposer de reprogrammer une nouvelle alarme.
6. Sinon :
    - Attendre une seconde puis recommencer la vérification.

✅ Challenges rencontrés et solutions apportées

1. Gestion du format de l'heure

Problème : L'utilisateur peut entrer un format incorrect (ex : 25:99:00).

Solution : Mettre en place une validation avec `try/except` ou une fonction de vérification du format.

2. Boucle infinie trop gourmande

Problème : Une boucle qui vérifie l'heure en continu peut consommer inutilement des ressources.

Solution : Ajouter `time.sleep(1)` pour réduire la charge CPU et vérifier l'heure seulement chaque seconde.

3. Déclenchement de l'alarme

Problème : Selon le système, jouer un son peut nécessiter des bibliothèques externes.

Solution :

- Utiliser un simple message texte pour une version basique.
- Ou installer une bibliothèque comme `playsound` pour jouer un fichier audio.

4. Gestion des erreurs utilisateur

Problème : Saisie vide, caractères non numériques, mauvais séparateurs…

Solution : Créer une boucle de validation qui redemande l'heure tant que le format n'est pas correct.

🚀 Comment développer ton programme d'alarme dans le futur

🎛️ 1. Créer une interface graphique (GUI)

Tu peux utiliser **Tkinter**, la bibliothèque graphique intégrée à Python, pour créer :

- Une fenêtre avec des boutons
- Un champ pour entrer l'heure
- Une liste des alarmes programmées
- Un bouton "Supprimer une alarme"

👉 Cela rend ton programme plus professionnel et plus agréable à utiliser.

🔊2. Ajouter un vrai son d'alarme

Au lieu d'un simple message texte, tu peux :

- Jouer un fichier audio (MP3, WAV)

- Utiliser la bibliothèque `playsound` ou `pygame`

☞ Cela donne un aspect plus réaliste à ton alarme.

🕒 3. Ajouter un compte à rebours

En plus de l'alarme à heure fixe, tu peux ajouter :

- Un minuteur (timer)
- Un chronomètre

☞ Cela élargit les fonctionnalités et t'apprend à gérer le temps différemment.

📱 4. Créer une version mobile ou web

Si tu veux aller plus loin :

- Version web avec **Flask** ou **Django**
- Version mobile avec **Kivy**

☞ Tu transformes ton petit programme en une vraie application utilisable partout.

💾 5. Sauvegarder les alarmes

Actuellement, les alarmes disparaissent quand tu fermes le programme. Tu peux les sauvegarder dans :

- Un fichier texte
- Un fichier JSON
- Une petite base de données SQLite

☞ Cela t'apprend la persistance des données.

🧠 6. Ajouter une intelligence simple

Tu peux rendre ton alarme plus "smart" :

- Détecter si l'heure entrée est déjà passée
- Proposer automatiquement le lendemain
- Ajouter des alarmes répétitives (tous les jours, tous les lundis, etc.)

☞ Tu commences à toucher à la logique avancée.

🔐 7. Créer une alarme de sécurité

Si tu veux aller vers la domotique :

- Détecter un mouvement (avec un capteur ou une webcam)
- Déclencher une alarme sonore
- Envoyer une notification

☞ C’est un excellent projet pour apprendre l’IoT.

✓ Conclusion

Le développement d’un programme d’alarme en Python permet de mettre en pratique des concepts essentiels comme la gestion du temps, les boucles et les conditions. Malgré quelques défis liés à la validation des entrées et à l’optimisation de la boucle de surveillance, les solutions adoptées permettent d’obtenir un programme fiable, simple et efficace. Ce projet constitue une base solide pour des extensions futures, telles que plusieurs alarmes, une interface graphique ou l’intégration dans un système domotique.
–capture d’ecrans du résultat :